Kernelized Bayesian Matrix Factorization

Mehmet Gönen[1] **Suleiman A. Khan**[1] **Samuel Kaski**[1,2]
[1] Helsinki Institute for Information Technology HIIT
Department of Information and Computer Science, Aalto University
[2] Helsinki Institute for Information Technology HIIT
Department of Computer Science, University of Helsinki

Abstract

We extend kernelized matrix factorization with a fully Bayesian treatment and with an ability to work with multiple side information sources expressed as different kernels. Kernel functions have been introduced to matrix factorization to integrate side information about the rows and columns (e.g., objects and users in recommender systems), which is necessary for making out-of-matrix (i.e., cold start) predictions. We discuss specifically bipartite graph inference, where the output matrix is binary, but extensions to more general matrices are straightforward. We extend the state of the art in two key aspects: (i) A fully conjugate probabilistic formulation of the kernelized matrix factorization problem enables an efficient variational approximation, whereas fully Bayesian treatments are not computationally feasible in the earlier approaches. (ii) Multiple side information sources are included, treated as different kernels in multiple kernel learning that additionally reveals which side information sources are informative. Our method outperforms alternatives in predicting drug–protein interactions on two data sets.

1 Introduction

Matrix factorization algorithms are very popular matrix completion methods (Srebro, 2004), successfully used in many applications including recommender systems and image inpainting. The main idea behind these methods is to factorize a partially observed data matrix by finding a low-dimensional latent representation for both its rows and columns. The prediction for a missing entry can be calculated as the inner product between the latent representations of the corresponding row and column. Salakhutdinov and Mnih (2008a,b) give a probabilistic formulation for matrix factorization and its fully Bayesian extension. However, these approaches are still incomplete in two major aspects: (i) It is not possible to integrate side information (e.g., social network or user profiles for recommender systems) into the model. (ii) It is not possible to make predictions for completely empty columns or rows (i.e., out-of-matrix prediction).

Algorithms for integrating side information into matrix factorization have been proposed earlier in the recommender systems literature. Ma et al. (2008) propose a probabilistic matrix factorization method that uses a social network and the rating matrix together to find better latent components. Shan and Banerjee (2010) integrate side information into a probabilistic matrix factorization model using topic models to generate latent components of the rated items (e.g., movies). Agarwal and Chen (2010) use a similar strategy to generate latent components of both users and items using topic models. Wang and Blei (2011) also combine matrix factorization and topic models for scientific article recommendation using textual content of articles as side information. All these algorithms are based on explicit feature representations; some are specific to count (e.g., text) data, and all are able to model linear dependencies. We use kernels to include nonlinear dependencies.

Recently, Zhou et al. (2012) propose a kernelized probabilistic matrix factorization method that generates latent components using Gaussian process priors with covariance matrices calculated on side information. However, with the modeling assumptions, only a *maximum a posteriori* (MAP) estimate for the latent components is computationally feasible, and even then the

used gradient descent approach can be very slow. Furthermore, the method uses only a single kernel for each domain and needs test instances while training to be able to calculate their latent components (i.e., *transductive learning*).

In this paper, we focus on modeling interaction networks between two domains (e.g., biological networks between drugs and proteins), and estimating unknown interactions between objects from these two domains, which is also known as *bipartite graph inference* (Yamanishi, 2009). The standard pairwise kernel approaches are based on a kernel matrix over object pairs in the training set and are computationally expensive (Ben-Hur and Noble, 2005). There are also kernel-based (non-Bayesian) dimensionality reduction algorithms that map objects from both domains into the same subspace and perform prediction there (Yamanishi, 2009; Yamanishi et al., 2008, 2010).

In biological interaction networks, being composed of structured objects such as drugs and proteins, there exist several feature representations or similarity measures for the objects (Schölkopf et al., 2004). Instead of using a single specific kernel, we can combine multiple kernel functions to obtain a better similarity measure, which is known as *multiple kernel learning* (Gönen and Alpaydin, 2011).

We introduce a kernelized Bayesian matrix factorization method and give its details for the bipartite graph inference scenario; it can also be applied to other types of matrices with slight modifications. Our two main contributions are: (i) We formulate a novel fully conjugate probabilistic model that allows us to develop an efficient variational approximation scheme, the first fully Bayesian treatment which is still significantly faster than the earlier method for computing MAP point estimates (Zhou et al., 2012). (ii) The proposed method is able to integrate multiple side information sources by coupling matrix factorization with multiple kernel learning. We show the effectiveness of our approach on one toy data set and two drug–protein interaction data sets.

2 Preliminaries and Notation

We assume that the objects come from two domains $\mathcal{X}$ and $\mathcal{Z}$. We are given two samples of independent and identically distributed training instances from each, denoted by $\mathbf{X} = \{\boldsymbol{x}_i \in \mathcal{X}\}_{i=1}^{N_\mathtt{x}}$ and $\mathbf{Z} = \{\boldsymbol{z}_j \in \mathcal{Z}\}_{j=1}^{N_\mathtt{z}}$. In order to calculate similarities between the objects of the same domain, we have multiple kernel functions for each domain, namely, $\{k_{\mathtt{x},m} \colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}\}_{m=1}^{P_\mathtt{x}}$ and $\{k_{\mathtt{z},n} \colon \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}\}_{n=1}^{P_\mathtt{z}}$. If the side information comes in the form of features instead of similarities, the set of kernels defined for a specific domain correspond to different notions of similarity on the same feature representation or may be using information coming from multiple feature representations (i.e., views).

The (i,j)th entry of the target label matrix $\mathbf{Y} \in \{-1,+1\}^{N_\mathtt{x} \times N_\mathtt{z}}$ is

$$y_j^i = \begin{cases} +1 & \text{if } \boldsymbol{x}_i \text{ and } \boldsymbol{z}_j \text{ are interacting,} \\ -1 & \text{otherwise.} \end{cases}$$

The superscript indexes the rows and the subscript indexes the columns. The prediction task is to estimate unknown interactions for out-of-matrix objects, which is also known as *cold start* prediction in recommender systems.

Figure 1 illustrates the method we propose; it is composed of four main parts: (a) kernel-based nonlinear dimensionality reduction, (b) multiple kernel learning, (c) matrix factorization, and (d) binary classification. The first two kernel-based parts are applied to each domain separately and they are completely symmetric, hence we call them *twins*. One of the twins (i.e., the one that operates on domain $\mathcal{Z}$) is omitted for clarity. In this section, we briefly explain each part and introduce the notation used. In the following sections, we formulate a fully conjugate probabilistic model and derive a variational approximation.

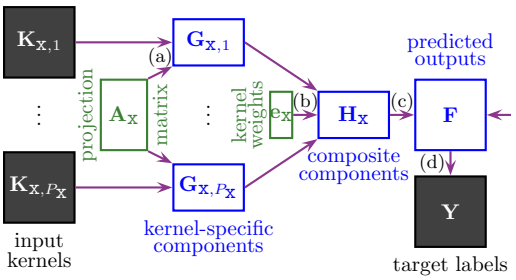

Figure 1: Flowchart of kernelized matrix factorization with twin multiple kernel learning (the twin domain $\mathcal{Z}$ is omitted for clarity).

2.1 Kernel-Based Nonlinear Dimensionality Reduction

In this part, we perform feature extraction using the input kernel matrices $\{\mathbf{K}_{\mathtt{x},m} \in \mathbb{R}^{N_\mathtt{x} \times N_\mathtt{x}}\}_{m=1}^{P_\mathtt{x}}$ and the common projection matrix $\mathbf{A}_\mathtt{x} \in \mathbb{R}^{N_\mathtt{x} \times R}$ where R is the corresponding subspace dimensionality. We obtain the kernel-specific components $\{\mathbf{G}_{\mathtt{x},m} = \mathbf{A}_\mathtt{x}^\top \mathbf{K}_{\mathtt{x},m}\}_{m=1}^{P_\mathtt{x}}$ after the projection. The main idea is very similar to *kernel principal component analysis* or *kernel Fisher discriminant analysis*, where the columns of the projection matrix can be solved with eigendecompositions (Schölkopf and Smola, 2002). However, this solution strategy is not possible for the more complex model formulated here.

2.2 Multiple Kernel Learning

This part is responsible of combining the kernel-specific components linearly to obtain the composite components $\mathbf{H}_{\mathtt{x}} = \sum_{m=1}^{P_{\mathtt{x}}} e_{\mathtt{x},m} \mathbf{G}_{\mathtt{x},m}$ where the kernel weights can take arbitrary values $\boldsymbol{e}_{\mathtt{x}} \in \mathbb{R}^{P_{\mathtt{x}}}$. If we have a single kernel function for a specific domain, we can safely ignore the composite components and the kernel weights, and use the single available kernel-specific components to represent the objects in that domain. The details of our method with a single kernel function for each domain are explained as the supplementary material in Appendix A.

2.3 Matrix Factorization

In this part, we propose to use the low-dimensional representations of objects in the unified subspace, namely, $\mathbf{H}_{\mathtt{x}}$ and $\mathbf{H}_{\mathtt{z}}$, to calculate the predicted output matrix $\mathbf{F} = \mathbf{H}_{\mathtt{x}}^{\top} \mathbf{H}_{\mathtt{z}}$. This corresponds to factorizing the predicted outputs into two low-rank matrices.

2.4 Binary Classification

This part just assigns a class label to each object pair $(\boldsymbol{x}_i, \boldsymbol{z}_j)$ by looking at the sign of the predicted output f_j^i in the matrix factorization part. The proposed method can also be extended to handle other types of outputs (e.g., real-valued outputs used in recommender systems) by removing the binary classification part and directly generating the target outputs in the matrix factorization part. This corresponds to removing the predicted output matrix $\mathbf{F}$ and generating target label matrix $\mathbf{Y}$ directly from the composite components $\mathbf{H}_{\mathtt{x}}$ and $\mathbf{H}_{\mathtt{z}}$. The details of our method for real-valued outputs are also given as the supplementary material in Appendix B.

3 Kernelized Bayesian Matrix Factorization with Twin Multiple Kernel Learning

For the method described in the previous section, we formulate a probabilistic model, called *kernelized Bayesian matrix factorization with twin multiple kernel learning* (KBMF2MKL), which has two key properties that enable us to perform efficient inference: (i) The kernel-specific and composite components are modeled explicitly by introducing them as latent variables. (ii) Kernel weights are assumed to be normally distributed without enforcing any constraints (e.g., non-negativity) on them. The reasons for introducing these two properties to our probabilistic model becomes clear when we explain our inference method.

Figure 2 gives the graphical model of KBMF2MKL with latent variables and their corresponding priors. There are some additions to the notation described earlier: The $N_{\mathtt{x}} \times R$ matrix of priors for the entries of the projection matrix $\mathbf{A}_{\mathtt{x}}$ is denoted by $\boldsymbol{\Lambda}_{\mathtt{x}}$. The $P_{\mathtt{x}} \times 1$ vector of priors for the kernel weights $\boldsymbol{e}_{\mathtt{x}}$ is denoted by $\boldsymbol{\eta}_{\mathtt{x}}$. The standard deviations for the kernel-specific and composite components are represented as σ_g and σ_h, respectively; these hyper-parameters are not shown for clarity.

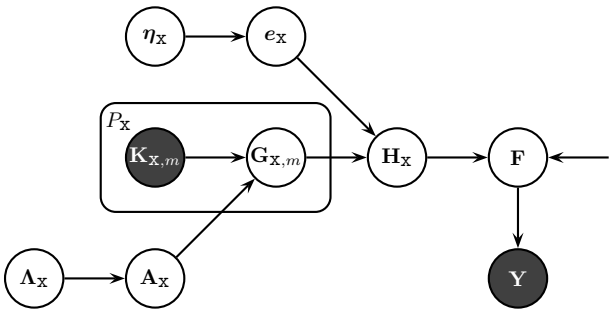

Figure 2: Graphical model of kernelized Bayesian matrix factorization with twin multiple kernel learning.

The distributional assumptions of the dimensionality reduction part are

$$\lambda_{\mathtt{x},s}^i \sim \mathcal{G}(\lambda_{\mathtt{x},s}^i; \alpha_\lambda, \beta_\lambda) \qquad \forall (i,s)$$
$$a_{\mathtt{x},s}^i | \lambda_{\mathtt{x},s}^i \sim \mathcal{N}(a_{\mathtt{x},s}^i; 0, (\lambda_{\mathtt{x},s}^i)^{-1}) \qquad \forall (i,s)$$
$$g_{\mathtt{x},m,i}^s | \boldsymbol{a}_{\mathtt{x},s}, \boldsymbol{k}_{\mathtt{x},m,i} \sim \mathcal{N}(g_{\mathtt{x},m,i}^s; \boldsymbol{a}_{\mathtt{x},s}^{\top} \boldsymbol{k}_{\mathtt{x},m,i}, \sigma_g^2) \qquad \forall (m,s,i)$$

where $\mathcal{N}(\cdot; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ is the normal distribution with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$, and $\mathcal{G}(\cdot; \alpha, \beta)$ denotes the gamma distribution with shape parameter α and scale parameter β. The multiple kernel learning part has the following distributional assumptions:

$$\eta_{\mathtt{x},m} \sim \mathcal{G}(\eta_{\mathtt{x},m}; \alpha_\eta, \beta_\eta) \qquad \forall m$$
$$e_{\mathtt{x},m} | \eta_{\mathtt{x},m} \sim \mathcal{N}(e_{\mathtt{x},m}; 0, \eta_{\mathtt{x},m}^{-1}) \qquad \forall m$$
$$h_{\mathtt{x},i}^s | \{e_{\mathtt{x},m}, g_{\mathtt{x},m,i}^s\}_{m=1}^{P_{\mathtt{x}}} \sim \mathcal{N}\left(h_{\mathtt{x},i}^s; \sum_{m=1}^{P_{\mathtt{x}}} e_{\mathtt{x},m} g_{\mathtt{x},m,i}^s, \sigma_h^2\right) \qquad \forall (s,i)$$

where kernel-level sparsity can be tuned by changing the hyper-parameters $(\alpha_\eta, \beta_\eta)$. Setting the gamma priors to induce sparsity, e.g., $(\alpha_\eta, \beta_\eta) = (0.001, 1000)$, produces results analogous to using the ℓ_1-norm on the kernel weights, whereas using uninformative priors, e.g., $(\alpha_\eta, \beta_\eta) = (1, 1)$, resembles using the ℓ_2-norm. The matrix factorization part calculates the predicted outputs using the inner products between the low-dimensional representations of the object pairs:

$$f_j^i | \boldsymbol{h}_{\mathtt{x},i}, \boldsymbol{h}_{\mathtt{z},j} \sim \mathcal{N}(f_j^i; \boldsymbol{h}_{\mathtt{x},i}^{\top} \boldsymbol{h}_{\mathtt{z},j}, 1) \qquad \forall (i,j)$$

where the predicted outputs are introduced to speed up the inference procedures (Albert and Chib, 1993).

The binary classification part forces the predicted outputs to have the same sign with their target labels:

$$y_j^i | f_j^i \sim \delta(f_j^i y_j^i > \nu) \qquad \forall (i,j)$$

where the margin parameter ν is introduced to remove ambiguity in the scaling and to place a low-density region between two classes, similar to the margin idea in SVMs, which is generally used for semi-supervised learning (Lawrence and Jordan, 2005). Here, $\delta(\cdot)$ is the Kronecker delta function that returns 1 if its argument is true and 0 otherwise.

4 Efficient Inference Using Variational Approximation

Exact inference for the model is intractable and of the two readily available alternatives, Gibbs sampling and variational approximation, we choose the latter for computational efficiency. Variational methods optimize a lower bound on the marginal likelihood, which involves a factorized approximation of the posterior, to find the joint parameter distribution (Beal, 2003).

As short-hand notations, all hyper-parameters in the model are denoted by $\boldsymbol{\zeta} = \{\alpha_\eta, \beta_\eta, \alpha_\lambda, \beta_\lambda, \sigma_g, \sigma_h, \nu\}$, all prior variables by $\boldsymbol{\Xi} = \{\boldsymbol{\eta}_\mathbf{x}, \boldsymbol{\eta}_\mathbf{z}, \boldsymbol{\Lambda}_\mathbf{x}, \boldsymbol{\Lambda}_\mathbf{z}\}$, and the remaining random variables by $\boldsymbol{\Theta} = \{\mathbf{A}_\mathbf{x}, \mathbf{A}_\mathbf{z}, \boldsymbol{e}_\mathbf{x}, \boldsymbol{e}_\mathbf{z}, \mathbf{F}, \{\mathbf{G}_{\mathbf{x},m}\}_{m=1}^{P_\mathbf{x}}, \{\mathbf{G}_{\mathbf{z},n}\}_{n=1}^{P_\mathbf{z}}, \mathbf{H}_\mathbf{x}, \mathbf{H}_\mathbf{z}\}$. We omit the dependence on $\boldsymbol{\zeta}$ for clarity. We factorize the variational approximation as

$$\begin{aligned} p(\boldsymbol{\Theta}, \boldsymbol{\Xi} | \{\mathbf{K}_{\mathbf{x},m}\}_{m=1}^{P_\mathbf{x}}, \{\mathbf{K}_{\mathbf{z},n}\}_{n=1}^{P_\mathbf{z}}, \mathbf{Y}) \approx q(\boldsymbol{\Theta}, \boldsymbol{\Xi}) = \\ q(\boldsymbol{\Lambda}_\mathbf{x}) q(\mathbf{A}_\mathbf{x}) q(\{\mathbf{G}_{\mathbf{x},m}\}_{m=1}^{P_\mathbf{x}}) q(\boldsymbol{\eta}_\mathbf{x}) q(\boldsymbol{e}_\mathbf{x}) q(\mathbf{H}_\mathbf{x}) \\ q(\boldsymbol{\Lambda}_\mathbf{z}) q(\mathbf{A}_\mathbf{z}) q(\{\mathbf{G}_{\mathbf{z},n}\}_{n=1}^{P_\mathbf{z}}) q(\boldsymbol{\eta}_\mathbf{z}) q(\boldsymbol{e}_\mathbf{z}) q(\mathbf{H}_\mathbf{z}) q(\mathbf{F}) \end{aligned}$$

and define each factor according to its full conditional:

$$\begin{aligned} q(\boldsymbol{\Lambda}_\mathbf{x}) &= \prod_{i=1}^{N_\mathbf{x}} \prod_{s=1}^{R} \mathcal{G}(\lambda_{\mathbf{x},s}^i; \alpha(\lambda_{\mathbf{x},s}^i), \beta(\lambda_{\mathbf{x},s}^i)) \\ q(\mathbf{A}_\mathbf{x}) &= \prod_{s=1}^{R} \mathcal{N}(\boldsymbol{a}_{\mathbf{x},s}; \mu(\boldsymbol{a}_{\mathbf{x},s}), \Sigma(\boldsymbol{a}_{\mathbf{x},s})) \\ q(\mathbf{G}_{\mathbf{x},m}) &= \prod_{i=1}^{N_\mathbf{x}} \mathcal{N}(\boldsymbol{g}_{\mathbf{x},m,i}; \mu(\boldsymbol{g}_{\mathbf{x},m,i}), \Sigma(\boldsymbol{g}_{\mathbf{x},m,i})) \qquad \forall m \\ q(\boldsymbol{e}_\mathbf{x}) &= \mathcal{N}(\boldsymbol{e}_\mathbf{x}; \mu(\boldsymbol{e}_\mathbf{x}), \Sigma(\boldsymbol{e}_\mathbf{x})) \\ q(\boldsymbol{\eta}_\mathbf{x}) &= \prod_{m=1}^{P_\mathbf{x}} \mathcal{G}(\eta_{\mathbf{x},m}; \alpha(\eta_{\mathbf{x},m}), \beta(\eta_{\mathbf{x},m})) \\ q(\mathbf{H}_\mathbf{x}) &= \prod_{i=1}^{N_\mathbf{x}} \mathcal{N}(\boldsymbol{h}_{\mathbf{x},i}; \mu(\boldsymbol{h}_{\mathbf{x},i}), \Sigma(\boldsymbol{h}_{\mathbf{x},i})) \\ q(\mathbf{F}) &= \prod_{i=1}^{N_\mathbf{x}} \prod_{j=1}^{N_\mathbf{z}} \mathcal{TN}(f_j^i; \mu(f_j^i), \Sigma(f_j^i), \rho(f_j^i)) \end{aligned}$$

where $\alpha(\cdot)$, $\beta(\cdot)$, $\mu(\cdot)$, and $\Sigma(\cdot)$ denote the shape parameter, scale parameter, mean vector, and covariance matrix, respectively. Here, $\mathcal{TN}(\cdot; \boldsymbol{\mu}, \boldsymbol{\Sigma}, \rho(\cdot))$ denotes the truncated normal distribution with mean vector $\boldsymbol{\mu}$, covariance matrix $\boldsymbol{\Sigma}$, and truncation rule $\rho(\cdot)$ such that $\mathcal{TN}(\cdot; \boldsymbol{\mu}, \boldsymbol{\Sigma}, \rho(\cdot)) \propto \mathcal{N}(\cdot; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ if $\rho(\cdot)$ is true and $\mathcal{TN}(\cdot; \boldsymbol{\mu}, \boldsymbol{\Sigma}, \rho(\cdot)) = 0$ otherwise.

We can bound the marginal likelihood using Jensen's inequality:

$$\begin{aligned} \log p(\mathbf{Y} | \{\mathbf{K}_{\mathbf{x},m}\}_{m=1}^{P_\mathbf{x}}, \{\mathbf{K}_{\mathbf{z},n}\}_{n=1}^{P_\mathbf{z}}) \geq \\ \mathrm{E}_{q(\boldsymbol{\Theta},\boldsymbol{\Xi})}[\log p(\mathbf{Y}, \boldsymbol{\Theta}, \boldsymbol{\Xi} | \{\mathbf{K}_{\mathbf{x},m}\}_{m=1}^{P_\mathbf{x}}, \{\mathbf{K}_{\mathbf{z},n}\}_{n=1}^{P_\mathbf{z}})] \\ - \mathrm{E}_{q(\boldsymbol{\Theta},\boldsymbol{\Xi})}[\log q(\boldsymbol{\Theta}, \boldsymbol{\Xi})] \end{aligned} \quad (1)$$

and optimize this bound by maximizing with respect to each factor separately until convergence. The approximate posterior distribution of a specific factor $\boldsymbol{\tau}$ can be found as

$$\begin{aligned} q(\boldsymbol{\tau}) \propto \\ \exp(\mathrm{E}_{q(\{\boldsymbol{\Theta},\boldsymbol{\Xi}\} \setminus \boldsymbol{\tau})}[\log p(\mathbf{Y}, \boldsymbol{\Theta}, \boldsymbol{\Xi} | \{\mathbf{K}_{\mathbf{x},m}\}_{m=1}^{P_\mathbf{x}}, \{\mathbf{K}_{\mathbf{z},n}\}_{n=1}^{P_\mathbf{z}})]). \end{aligned}$$

For our model, thanks to the conjugacy, the resulting approximate posterior distribution of each factor follows the same distribution as the corresponding factor.

The approximate posterior distributions of the dimensionality reduction part can be found as

$$\begin{aligned} q(\boldsymbol{\Lambda}_\mathbf{x}) &= \prod_{i=1}^{N_\mathbf{x}} \prod_{s=1}^{R} \mathcal{G}\left(\lambda_{\mathbf{x},s}^i; \alpha_\lambda + \frac{1}{2}, \left(\frac{1}{\beta_\lambda} + \frac{\widetilde{(a_{\mathbf{x},s}^i)^2}}{2}\right)^{-1}\right) \\ q(\mathbf{A}_\mathbf{x}) &= \prod_{s=1}^{R} \mathcal{N}\Bigg(\boldsymbol{a}_{\mathbf{x},s}; \Sigma(\boldsymbol{a}_{\mathbf{x},s}) \sum_{m=1}^{P_\mathbf{x}} \frac{\mathbf{K}_{\mathbf{x},m} \widetilde{(\boldsymbol{g}_{\mathbf{x},m}^s)}^\top}{\sigma_g^2}, \\ &\qquad \left(\mathrm{diag}(\widetilde{\boldsymbol{\lambda}_\mathbf{x}^s}) + \sum_{m=1}^{P_\mathbf{x}} \frac{\mathbf{K}_{\mathbf{x},m} \mathbf{K}_{\mathbf{x},m}^\top}{\sigma_g^2}\right)^{-1}\Bigg) \end{aligned} \quad (2)$$

where the tilde notation denotes the posterior expectations as usual, i.e., $\widetilde{f(\boldsymbol{\tau})} = \mathrm{E}_{q(\boldsymbol{\tau})}[f(\boldsymbol{\tau})]$.

The kernel-specific components have the following approximate posterior distribution:

$$\begin{aligned} q(\mathbf{G}_{\mathbf{x},m}) = \prod_{i=1}^{N_\mathbf{x}} \mathcal{N}\Big(\boldsymbol{g}_{\mathbf{x},m,i}; \Sigma(\boldsymbol{g}_{\mathbf{x},m,i}) \\ \left(\frac{\widetilde{\mathbf{A}_\mathbf{x}^\top} \boldsymbol{k}_{\mathbf{x},m,i}}{\sigma_g^2} + \frac{\widetilde{e_{\mathbf{x},m}} \widetilde{\boldsymbol{h}_{\mathbf{x},i}}}{\sigma_h^2} - \sum_{o \neq m} \frac{\widetilde{e_{\mathbf{x},m} e_{\mathbf{x},o}} \widetilde{\boldsymbol{g}_{\mathbf{x},o,i}}}{\sigma_h^2}\right), \\ \left(\frac{\mathbf{I}}{\sigma_g^2} + \frac{\widetilde{e_{\mathbf{x},m}^2} \mathbf{I}}{\sigma_h^2}\right)^{-1}\Big) \end{aligned} \quad \forall m \quad (3)$$

where the mean and covariance parameters are affected by the kernel weights, the composite components, and other kernel-specific components in addition to the projection matrix and the corresponding kernel matrix.

The approximate posterior distributions of the multiple kernel learning part can be found as

$$q(\boldsymbol{\eta}_{\mathbf{x}}) = \prod_{m=1}^{P_{\mathbf{x}}} \mathcal{G}\left(\eta_{\mathbf{x},m}; \alpha_\eta + \frac{1}{2}, \left(\frac{1}{\beta_\eta} + \frac{\widetilde{e^2_{\mathbf{x},m}}}{2}\right)^{-1}\right)$$

$$q(\boldsymbol{e}_{\mathbf{x}}) = \mathcal{N}\left(\boldsymbol{e}_{\mathbf{x}}; \Sigma(\boldsymbol{e}_{\mathbf{x}}) \left[\frac{\widetilde{\mathbf{G}^\top_{\mathbf{x},m}}\widetilde{\mathbf{H}_{\mathbf{x}}}}{\sigma_h^2}\right]_{m=1}^{P_{\mathbf{x}}}, \left(\operatorname{diag}(\widetilde{\boldsymbol{\eta}_{\mathbf{x}}}) + \left[\frac{\widetilde{\mathbf{G}^\top_{\mathbf{x},m}\mathbf{G}_{\mathbf{x},o}}}{\sigma_h^2}\right]_{m=1,o=1}^{P_{\mathbf{x}},P_{\mathbf{x}}}\right)^{-1}\right) \quad (4)$$

where the mean and covariance parameters of the kernel weights are calculated using the kernel-specific and composite components.

The composite components have the following approximate posterior distribution:

$$q(\mathbf{H}_{\mathbf{x}}) = \prod_{i=1}^{N_{\mathbf{x}}} \mathcal{N}\left(\boldsymbol{h}_{\mathbf{x},i}; \Sigma(\boldsymbol{h}_{\mathbf{x},i}) \left(\sum_{m=1}^{P_{\mathbf{x}}} \frac{\widetilde{e_{\mathbf{x},m}}\widetilde{\boldsymbol{g}_{\mathbf{x},m,i}}}{\sigma_h^2} + \widetilde{\mathbf{H}_{\mathbf{z}}}\widetilde{(\boldsymbol{f}^i)^\top}\right), \left(\frac{\mathbf{I}}{\sigma_h^2} + \widetilde{\mathbf{H}_{\mathbf{z}}\mathbf{H}_{\mathbf{z}}^\top}\right)^{-1}\right) \quad (5)$$

where it can be seen that the inference transfers information between the two domains. Note that the composite components of each domain are the only random variables that have an effect on the other domain, i.e., only the $\mathbf{H}_{\mathbf{z}}$ variables of domain $\mathcal{Z}$ are used when updating the random variables of the domain $\mathcal{X}$.

The approximate posterior distribution of the predicted outputs is a product of truncated normals:

$$q(\mathbf{F}) = \prod_{i=1}^{N_{\mathbf{x}}} \prod_{j=1}^{N_{\mathbf{z}}} \mathcal{TN}(f^i_j; \widetilde{\boldsymbol{h}^\top_{\mathbf{x},i}}\widetilde{\boldsymbol{h}_{\mathbf{z},j}}, 1, f^i_j y^i_j > \nu). \quad (6)$$

We need to find the posterior expectation of $\mathbf{F}$ to update the approximate posterior distributions of the composite components. Fortunately, the truncated normal distribution has a closed-form formula for its expectation.

4.1 Modeling Choices

Note that using the kernel-specific and composite components as latent variables in our probabilistic model introduces extra conditional independencies between the random variables and enables us to derive efficient update rules for the approximate posterior distributions. The other key property of our model is the assumption of normality of the kernel weights, which allows us to obtain a fully conjugate probabilistic model (Gönen, 2012). In earlier Bayesian multiple kernel learning algorithms, the combined kernel has usually been defined as a convex sum of the input kernels, by assuming a Dirichlet distribution on the kernel weights (Girolami and Rogers, 2005; Damoulas and Girolami, 2008). As a consequence of the nonconjugacy between Dirichlet and normal distributions, they need to use a sampling strategy (e.g., importance sampling) to update the kernel weights when deriving variational approximations.

4.2 Convergence

The inference mechanism sequentially updates the approximate posterior distributions of the latent variables and the corresponding priors until convergence, which can be checked by monitoring the lower bound in (1). The first term of the lower bound corresponds to the sum of exponential-form expectations of the distributions in the joint likelihood. The second term is the sum of negative entropies of the approximate posteriors in the ensemble. The only nonstandard distribution in these terms is the truncated normal distribution used for the predicted outputs, and the truncated normal distribution has a closed-form formula also for its entropy.

4.3 Computational Complexity

The most time-consuming operations of the update equations are covariance calculations because they need matrix inversions. The time complexity of the covariance updates for the projection matrices in (2) is $\mathcal{O}(R\max(N^3_{\mathbf{x}}, N^3_{\mathbf{z}}))$ and we can cache $\sum_{m=1}^{P_{\mathbf{x}}} \mathbf{K}_{\mathbf{x},m}\mathbf{K}^\top_{\mathbf{x},m}$ and $\sum_{n=1}^{P_{\mathbf{z}}} \mathbf{K}_{\mathbf{z},n}\mathbf{K}^\top_{\mathbf{z},n}$ before starting inference procedure to reduce the computational cost of these updates. The covariance updates for the kernel-specific components in (3) require inverting diagonal matrices. The time complexities of the covariance updates for the kernel weights and the composite components in (4) and (5) are $\mathcal{O}(\max(P^3_{\mathbf{x}}, P^3_{\mathbf{z}}))$. The other calculations in these updates can be done efficiently using matrix-matrix or matrix-vector multiplications. Finding the posterior expectations of the predicted outputs in (6) only requires evaluating the standardized normal cumulative distribution function and the standardized normal probability density. In summary, the total time complexity of each iteration in our variational approximation scheme is $\mathcal{O}(R\max(N^3_{\mathbf{x}}, N^3_{\mathbf{z}}) + \max(P^3_{\mathbf{x}}, P^3_{\mathbf{z}}))$, which makes the

algorithm more efficient than standard pairwise kernel approaches (Ben-Hur and Noble, 2005) that require calculating a kernel matrix over pairs and training a kernel-based classifier using this kernel, resulting in $\mathcal{O}(N_{\mathbf{x}}^3 N_{\mathbf{z}}^3)$ complexity.

4.4 Prediction

Given a test pair $(\boldsymbol{x}_\star, \boldsymbol{z}_*)$, we want to predict the corresponding score $f_*^\star$ or target label $y_*^\star$. We first replace posterior distributions of $\mathbf{A}_{\mathbf{x}}$, $\mathbf{A}_{\mathbf{z}}$, $\boldsymbol{e}_{\mathbf{x}}$, and $\boldsymbol{e}_{\mathbf{z}}$ with their approximate posterior distributions $q(\mathbf{A}_{\mathbf{x}})$, $q(\mathbf{A}_{\mathbf{z}})$, $q(\boldsymbol{e}_{\mathbf{x}})$, and $q(\boldsymbol{e}_{\mathbf{z}})$. Using the approximate distributions, we obtain the predictive distributions of the kernel-specific and composite components. The predictive distribution of the target label can finally be formulated as

$$p(y_*^\star = +1 | \{\boldsymbol{k}_{\mathbf{x},m,\star}, \mathbf{K}_{\mathbf{x},m}\}_{m=1}^{P_{\mathbf{x}}}, \{\boldsymbol{k}_{\mathbf{z},n,*}, \mathbf{K}_{\mathbf{z},n}\}_{n=1}^{P_{\mathbf{z}}}, \mathbf{Y}) = (\mathcal{Z}_*^\star)^{-1} \Phi\left(\frac{\mu(f_*^\star) - \nu}{\Sigma(f_*^\star)}\right)$$

where $\mathcal{Z}_*^\star$ is the normalization coefficient calculated for the test pair and $\Phi(\cdot)$ is the standardized normal cumulative distribution function.

5 Experiments

We first run our method on a toy data set to illustrate its kernel learning capability. We then test its performance in a real-life application with experiments on two drug–protein interaction data sets. One of them is a standard data set with a single view for each domain and the other one is a larger multiview data set we have collected. Our Matlab implementations will be available at `http://research.ics.aalto.fi/mi/software/kbmf2mkl/`.

5.1 Toy Data Set

We create a toy data set consisting of samples from two domains and real-valued outputs for object pairs. The data generation process is:

$$\begin{aligned} x_i^m &\sim \mathcal{N}(x_i^m; 0, 1) && \forall(m, i) \\ z_j^n &\sim \mathcal{N}(z_j^n; 0, 1) && \forall(n, j) \\ y_j^i | \boldsymbol{x}_i, \boldsymbol{z}_j &\sim \mathcal{N}(y_j^i; x_i^1 z_j^3 + x_i^4 z_j^8 + x_i^7 z_j^{10}, 1) && \forall(i, j) \end{aligned}$$

where $(N_{\mathbf{x}}, N_{\mathbf{z}}) = (40, 60)$, the samples from $\mathcal{X}$ and $\mathcal{Z}$ are generated from 15- and 10-dimensional isotropic normal distributions with unit variance (i.e., $m \in \{1, \ldots, 15\}$ and $n \in \{1, \ldots, 10\}$), respectively, and the target outputs are generated using only three features from each domain. Note that this data set has not been generated from our probabilistic model.

In order to have multiple kernel functions for each domain, we calculate a separate linear kernel for each feature of the data points, i.e., $(P_{\mathbf{x}}, P_{\mathbf{z}}) = (15, 10)$. We then learn our model, intended to work as a predictive model that identifies the relevant features for the prediction task and has a good generalization performance. We use uninformative priors for the projection matrices and the kernel weights by setting $(\alpha_\eta, \beta_\eta, \alpha_\lambda, \beta_\lambda) = (1, 1, 1, 1)$. The standard deviations are set to $(\sigma_g, \sigma_h, \sigma_y) = (0.1, 0.1, 1)$. The subspace dimensionality is arbitrarily set to five (i.e., $R = 5$).

Figure 3 shows the found posterior means of the kernel weights. We see that our method correctly identifies the relevant features for each domain (i.e., the first, fourth, and seventh features for $\mathcal{X}$ and the third, eighth, and tenth features for $\mathcal{Z}$). The *root mean square error* between the target and predicted outputs is 0.9865 in accordance with the level of noise added.

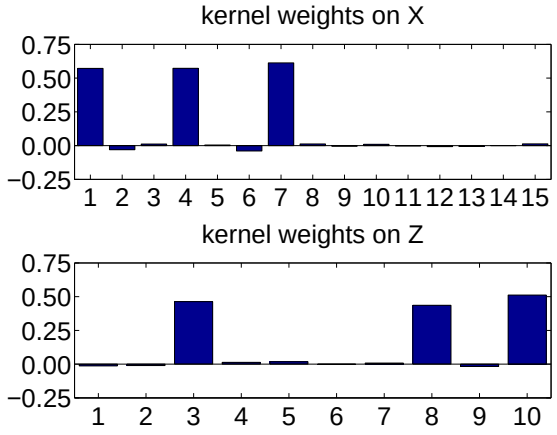

Figure 3: Posterior means of the kernel weights found by our method on the toy data set.

5.2 Drug–Protein Data Set of Yamanishi et al. (2008)

As the first case study, we analyze a drug–protein interaction network of humans, involving enzymes in particular. This drug–protein interaction network contains 445 drugs, 664 proteins, and 2926 experimentally validated interactions between them. The data set consists of the chemical similarity matrix between drugs, the genomic similarity matrix between proteins, and the target matrix of known interactions provided by Yamanishi et al. (2008).

We compare one baseline and three matrix factorization methods: (i) `Baseline` simply calculates the target output averages over each column or row as the predictions, (ii) *kernelized probabilistic matrix factorization* (`KPMF`) method of Zhou et al. (2012) with real-valued outputs, (iii) our *kernelized probabilistic matrix*

factorization (KBMF) method with real-valued outputs, and (iv) KBMF method with binary outputs.

Our experimental methodology is as follows: For KPMF, the standard deviation σ_y is set to one. For both KBMF variants, we use uninformative priors for the projection matrices and the kernel weights, i.e., $(\alpha_\eta, \beta_\eta, \alpha_\lambda, \beta_\lambda) = (1, 1, 1, 1)$, and the standard deviations (σ_g, σ_h) are set to $(0.1, 0.1)$. For KBMF with real-valued outputs, the standard deviation σ_y is set to one. For KBMF with binary outputs, the margin parameter ν is arbitrarily set to one. We perform simulations with eight different numbers of components, i.e., $R \in \{5, 10, \ldots, 40\}$. We run five replications of five-fold cross validation over drugs and report the average *area under ROC curve* (AUC) over the 25 results as the performance measure.

In the results, C and G mark the chemical similarity between drugs and the genomic similarity between proteins, respectively, whereas N marks the similarity between proteins calculated from the interaction network and it is defined as the ratio between (i) the number of drugs that are interacting with both proteins and (ii) the number of drugs that are interacting with at least one of the proteins, (i.e., Jaccard index).

The results in Figure 4 reveal that KPMF is above the baseline for more than 5 components, and both variants of KBMF for all component numbers. Both variants of our new KBMF outperform the earlier KPMF for all types of inputs. The difference is not due to KPMF having been introduced only for real-valued outputs, as even the real-output variant of KBMF is better. The difference is not due to the inability of the current version of KPMF to handle multiple data views either, as

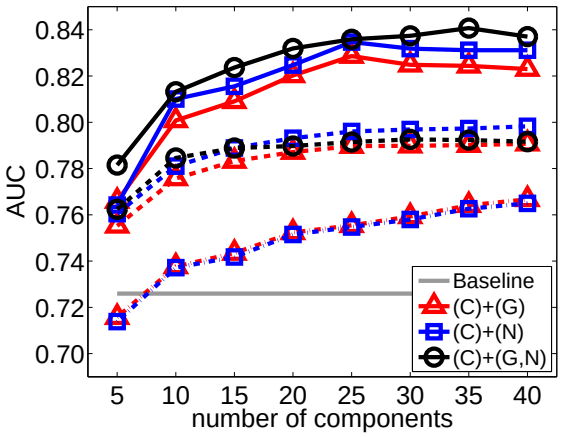

Figure 4: Average prediction performances (area under ROC curve, AUC) on the drug–protein data set of Yamanishi et al. (2008). Gray solid line: Baseline; other solid lines: KBMF with binary outputs; dashed lines: KBMF; dash-dotted lines: KPMF.

the single-kernel KBMF outperforms it. Hence the differences in the performance are due to the differences in the inference: MAP point estimates versus fully Bayesian inference. The best results are obtained with the binary-output KBMF when using all data sources.

Note that when we combine the genomic and network similarities between proteins using our method, the resulting similarity measure for proteins is better than those of single-kernel scenarios, leading to better prediction performance. This shows that when we have multiple side information sources about the objects, integrating them into the matrix factorization model in a principled way improves the results.

5.3 Drug–Protein Data Set of Khan et al. (2012)

We study an additional drug–protein interaction network of humans, containing 855 drugs, 800 proteins, and 4659 experimentally validated interactions between them, extracted from the drugs and proteins of the data set provided by Khan et al. (2012). We have two views consisting of two standard 3D chemical structure descriptors for drugs, namely, 1120-dimensional *Amanda* (Duran et al., 2008) and 76-dimensional *VolSurf* (Cruciani et al., 2000). In order to calculate the similarity between drugs, we use a Gaussian kernel whose width parameter is selected as the square root of the dimensionality of the data points.

We repeat the same experimental procedure as in the previous experiment with one minor change only. We perform simulations with 16 different numbers of components, i.e., $R \in \{5, 10, \ldots, 80\}$, due to the larger size of the interaction network.

We compare four different ways of including the drug property data. Amanda and VolSurf correspond to using a single view when calculating the kernel between drugs. Early corresponds to concatenating the two views, which is known as *early combination* (Schölkopf et al., 2004), before calculating the kernel between drugs. MKL corresponds to calculating two different kernels between drugs and combining them with our kernel combination approach.

The overall ordering of the results of the different matrix factorization methods is the same as in the previous case study (Figure 5). The KPMF outperforms Baseline after 20 components, whereas KBMF is consistently and significantly better (by at least four percentage units) than KPMF for all single-kernel scenarios. KBMF with five components is already better than Baseline for all scenarios. We do not report the results of KBMF with real-valued outputs in order not to clutter the figure.

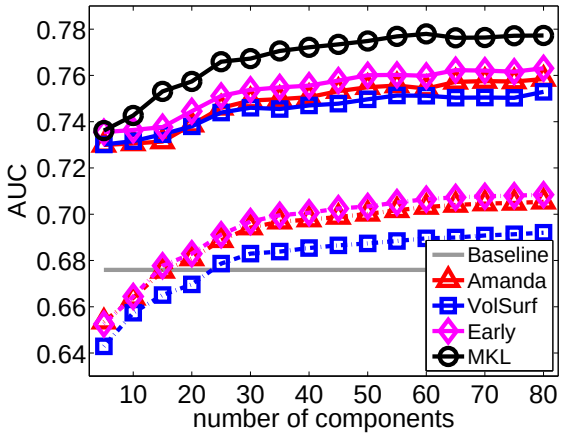

Figure 5: Average prediction performances (area under ROC curve, AUC) on the drug–protein data set of Khan et al. (2012). Gray solid line: `Baseline`; solid lines: `KBMF` with binary outputs; dash-dotted lines `KPMF`.

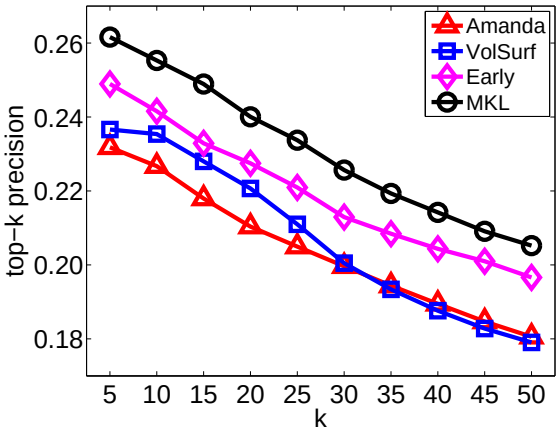

Figure 6: Average precision (over query drugs) of retrieval as a function of number k of retrieved drugs. See the text for details.

For `KBMF` with binary outputs, we see that `Amanda` and `VolSurf` are significantly better than `Baseline` and obtain similar prediction performances. `Early` outperforms `Amanda` and `VolSurf` with a slight margin, whereas `MKL` obtains consistently better results than all the other scenarios after five components.

Our method can also be interpreted as a metric learning algorithm since each kernel function can be converted into a distance metric. We test this property on the task of finding or *retrieving* drugs with similar functions. The idea is that since the targets are centrally important for the action mechanisms of drugs, a metric useful for predicting targets could be useful for retrieval of drugs as well. As the ground truth for relevance we use a standard therapeutic classification of the drugs according to the organ or system on which they act and/or their chemical characteristics (not used during learning); drugs having the same class are considered relevant. Figure 6 gives the precision at top-k retrieved drugs, when each drug in turn is used as the query and the rest of the 855 drugs are retrieved in the order of similarity according to the learned metric. `Early` is better than `Amanda` and `VolSurf`, and `MKL` is the best. This shows that our method is able to learn a kernel function between drugs that is better for retrieval than the kernels either on single or concatenated views.

6 Discussion

We introduce a kernelized Bayesian matrix factorization method that can make use of multiple side information sources about the objects (both rows and columns) and be applied in various scenarios including interaction network modeling and recommender systems. Our two main contributions are: (i) formulating an efficient variational approximation scheme for inference with the help of a novel fully conjugate probabilistic model and (ii) coupling matrix factorization with multiple kernel learning to integrate multiple side information sources into the model. In contrast to the earlier kernelized probabilistic matrix factorization method of Zhou et al. (2012), for our probabilistic model, it is possible to derive a computationally feasible fully Bayesian treatment. We illustrate the usefulness of the method on one toy data set and two molecular biological data sets.

An interesting topic for future research is to optimize the dimensionality of the latent components using a Bayesian model selection procedure. For example, we can share the same set of precision priors for the projection matrices and determine the dimensionality using *automatic relevance determination* (Neal, 1996).

Acknowledgments. This work was financially supported by the Academy of Finland (Finnish Centre of Excellence in Computational Inference Research COIN, grant no 251170; Computational Modeling of the Biological Effects of Chemicals, grant no 140057) and the Finnish Graduate School in Computational Sciences (FICS).

References

D. Agarwal and B.-C. Chen. fLDA: Matrix factorization through latent Dirichlet allocation. In *Proceedings of the 3rd ACM International Conference on Web Search and Data Mining*, pages 91–100, 2010.

J. H. Albert and S. Chib. Bayesian analysis of binary

and polychotomous response data. *Journal of the American Statistical Association*, 88(422):669–679, 1993.

M. J. Beal. *Variational Algorithms for Approximate Bayesian Inference*. PhD thesis, The Gatsby Computational Neuroscience Unit, University College London, 2003.

A. Ben-Hur and W. S. Noble. Kernel methods for predicting protein–protein interactions. *Bioinformatics*, 21(Suppl. 1):i38–i46, 2005.

G. Cruciani, M. Pastor, and W. Guba. VolSurf: A new tool for the pharmacokinetic optimization of lead compounds. *European Journal of Pharmaceutical Sciences*, 11(Suppl. 2):S29–S39, 2000.

T. Damoulas and M. A. Girolami. Probabilistic multi-class multi-kernel learning: On protein fold recognition and remote homology detection. *Bioinformatics*, 24(10):1264–1270, 2008.

A. Duran, G. C. Martinez, and M. Pastor. Development and validation of AMANDA, a new algorithm for selecting highly relevant regions in Molecular Interaction Fields. *Journal of Chemical Information and Modeling*, 48(9):1813–1823, 2008.

M. Girolami and S. Rogers. Hierarchic Bayesian models for kernel learning. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 241–248, 2005.

M. Gönen. Bayesian efficient multiple kernel learning. In *Proceedings of the 29th International Conference on Machine Learning*, pages 1–8, 2012.

M. Gönen and E. Alpaydin. Multiple kernel learning algorithms. *Journal of Machine Learning Research*, 12(Jul):2211–2268, 2011.

S. A. Khan, A. Faisal, J. P. Mpindi, J. A. Parkkinen, T. Kalliokoski, A. Poso, O. P. Kallioniemi, K. Wennerberg, and S. Kaski. Comprehensive data-driven analysis of the impact of chemoinformatic structure on the genome-wide biological response profiles of cancer cells to 1159 drugs. *BMC Bioinformatics*, 13 (112), 2012.

N. D. Lawrence and M. I. Jordan. Semi-supervised learning via Gaussian processes. In *Advances in Neural Information Processing Systems 17*, pages 753–760, 2005.

H. Ma, H. Yang, M. R. Lyu, and I. King. SoRec: Social recommendation using probabilistic matrix factorization. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, pages 931–940, 2008.

R. M. Neal. *Bayesian Learning for Neural Networks*. Springer, New York, NY, 1996.

R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems 20*, pages 1257–1264, 2008a.

R. Salakhutdinov and A. Mnih. Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In *Proceedings of the 25nd International Conference on Machine Learning*, pages 880–887, 2008b.

B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, 2002.

B. Schölkopf, K. Tsuda, and J.-P. Vert, editors. *Kernel Methods in Computational Biology*. MIT Press, Cambridge, MA, 2004.

H. Shan and A. Banerjee. Generalized probabilistic matrix factorizations for collaborative filtering. In *Proceedings of the 2010 IEEE International Conference on Data Mining*, pages 1025–1030, 2010.

N. Srebro. *Learning with Matrix Factorizations*. PhD thesis, Massachusetts Institute of Technology, 2004.

C. Wang and D. M. Blei. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 448–456, 2011.

Y. Yamanishi. Supervised bipartite graph inference. In *Advances in Neural Information Processing Systems 21*, pages 1841–1848, 2009.

Y. Yamanishi, M. Araki, A. Gutteridge, W. Honda, and M. Kaneisha. Prediction of drug–target interaction networks from the integration of chemical and genomic spaces. *Bioinformatics*, 24:i232–i240, 2008.

Y. Yamanishi, M. Kotera, M. Kanesiha, and S. Goto. Drug–target interaction prediction from chemical, genomic and pharmacological data in an integrated framework. *Bioinformatics*, 26:i246–i254, 2010.

T. Zhou, H. Shan, A. Banerjee, and G. Sapiro. Kernelized probabilistic matrix factorization: Exploiting graphs and side information. In *Proceedings of the 2012 SIAM International Conference on Data Mining*, pages 403–414, 2012.

Supplementary Material

In this supplementary material, we provide details about two variants of our method: (A) for a single kernel function on each domain instead of the multiple kernels, and (B) for real-valued outputs instead of the binary outputs.

A Kernelized Bayesian Matrix Factorization with Twin Kernels

We formulate a simplified probabilistic model, called *kernelized Bayesian matrix factorization with twin kernels* (KBMF2K), for the case with a single kernel function for each domain. Figure 7 shows the graphical model of KBMF2K with latent variables and their corresponding priors.

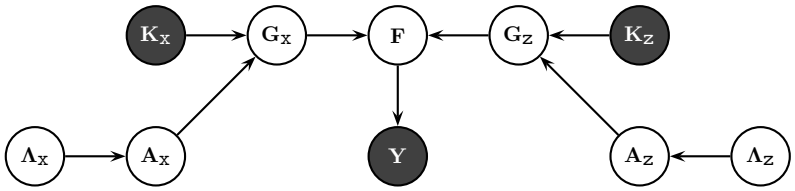

Figure 7: Graphical model of kernelized Bayesian matrix factorization with twin kernels.

The distributional assumptions of the simplified model are

$$\begin{aligned}
\lambda^i_{\mathbf{x},s} &\sim \mathcal{G}(\lambda^i_{\mathbf{x},s}; \alpha_\lambda, \beta_\lambda) && \forall(i,s)\\
a^i_{\mathbf{x},s}|\lambda^i_{\mathbf{x},s} &\sim \mathcal{N}(a^i_{\mathbf{x},s}; 0, (\lambda^i_{\mathbf{x},s})^{-1}) && \forall(i,s)\\
g^s_{\mathbf{x},i}|\boldsymbol{a}_{\mathbf{x},s}, \boldsymbol{k}_{\mathbf{x},i} &\sim \mathcal{N}(g^s_{\mathbf{x},i}; \boldsymbol{a}^\top_{\mathbf{x},s}\boldsymbol{k}_{\mathbf{x},i}, \sigma_g^2) && \forall(s,i)\\
f^i_j|\boldsymbol{g}_{\mathbf{x},i}, \boldsymbol{g}_{\mathbf{z},j} &\sim \mathcal{N}(f^i_j; \boldsymbol{g}^\top_{\mathbf{x},i}\boldsymbol{g}_{\mathbf{z},j}, 1) && \forall(i,j)\\
y^i_j|f^i_j &\sim \delta(y^i_j; f^i_j y^i_j > \nu) && \forall(i,j).
\end{aligned}$$

As short-hand notations, all hyper-parameters in the model are denoted by $\boldsymbol{\zeta} = \{\alpha_\lambda, \beta_\lambda, \sigma_g, \nu\}$, all prior variables by $\mathbf{\Xi} = \{\mathbf{\Lambda}_\mathbf{x}, \mathbf{\Lambda}_\mathbf{z}\}$, and the remaining random variables by $\mathbf{\Theta} = \{\mathbf{A}_\mathbf{x}, \mathbf{A}_\mathbf{z}, \mathbf{F}, \mathbf{G}_\mathbf{x}, \mathbf{G}_\mathbf{z}\}$. We again omit the dependence on $\boldsymbol{\zeta}$ for clarity. We can write the factorized variational approximation as

$$p(\mathbf{\Theta}, \mathbf{\Xi}|\mathbf{K}_\mathbf{x}, \mathbf{K}_\mathbf{z}, \mathbf{Y}) \approx q(\mathbf{\Theta}, \mathbf{\Xi}) = q(\mathbf{\Lambda}_\mathbf{x})q(\mathbf{A}_\mathbf{x})q(\mathbf{G}_\mathbf{x})q(\mathbf{\Lambda}_\mathbf{z})q(\mathbf{A}_\mathbf{z})q(\mathbf{G}_\mathbf{z})q(\mathbf{F})$$

and define each factor in the ensemble just like its full conditional:

$$\begin{aligned}
q(\mathbf{\Lambda}_\mathbf{x}) &= \prod_{i=1}^{N_\mathbf{x}}\prod_{s=1}^{R} \mathcal{G}(\lambda^i_{\mathbf{x},s}; \alpha(\lambda^i_{\mathbf{x},s}), \beta(\lambda^i_{\mathbf{x},s}))\\
q(\mathbf{A}_\mathbf{x}) &= \prod_{s=1}^{R} \mathcal{N}(\boldsymbol{a}_{\mathbf{x},s}; \mu(\boldsymbol{a}_{\mathbf{x},s}), \Sigma(\boldsymbol{a}_{\mathbf{x},s}))\\
q(\mathbf{G}_\mathbf{x}) &= \prod_{m=1}^{P_\mathbf{x}}\prod_{i=1}^{N_\mathbf{x}} \mathcal{N}(\boldsymbol{g}_{\mathbf{x},i}; \mu(\boldsymbol{g}_{\mathbf{x},i}), \Sigma(\boldsymbol{g}_{\mathbf{x},i}))\\
q(\mathbf{F}) &= \prod_{i=1}^{N_\mathbf{x}}\prod_{j=1}^{N_\mathbf{z}} \mathcal{TN}(f^i_j; \mu(f^i_j), \Sigma(f^i_j), \rho(f^i_j)).
\end{aligned}$$

We can bound the marginal likelihood using Jensen's inequality:

$$\log p(\mathbf{Y}|\mathbf{K}_\mathbf{x}, \mathbf{K}_\mathbf{z}) \geq \mathrm{E}_{q(\mathbf{\Theta},\mathbf{\Xi})}[\log p(\mathbf{Y}, \mathbf{\Theta}, \mathbf{\Xi}|\mathbf{K}_\mathbf{x}, \mathbf{K}_\mathbf{z})] - \mathrm{E}_{q(\mathbf{\Theta},\mathbf{\Xi})}[\log q(\mathbf{\Theta}, \mathbf{\Xi})]$$

and optimize this bound by maximizing with respect to each factor separately until convergence. The approximate posterior distribution of a specific factor $\boldsymbol{\tau}$ can be found as

$$q(\boldsymbol{\tau}) \propto \exp(\mathrm{E}_{q(\{\mathbf{\Theta},\mathbf{\Xi}\}\setminus\boldsymbol{\tau})}[\log p(\mathbf{Y}, \mathbf{\Theta}, \mathbf{\Xi}|\mathbf{K}_\mathbf{x}, \mathbf{K}_\mathbf{z})]).$$

The approximate posterior distributions of the ensemble can be found as

$$q(\boldsymbol{\Lambda}_{\mathbf{x}}) = \prod_{i=1}^{N_{\mathbf{x}}} \prod_{s=1}^{R} \mathcal{G}\left(\lambda_{\mathbf{x},s}^{i}; \alpha_{\lambda} + \frac{1}{2}, \left(\frac{1}{\beta_{\lambda}} + \frac{\widetilde{(a_{\mathbf{x},s}^{i})^{2}}}{2}\right)^{-1}\right)$$

$$q(\mathbf{A}_{\mathbf{x}}) = \prod_{s=1}^{R} \mathcal{N}\left(\boldsymbol{a}_{\mathbf{x},s}; \Sigma(\boldsymbol{a}_{\mathbf{x},s}) \frac{\mathbf{K}_{\mathbf{x}} \widetilde{(\boldsymbol{g}_{\mathbf{x}}^{s})^{\top}}}{\sigma_{g}^{2}}, \left(\operatorname{diag}(\widetilde{\boldsymbol{\lambda}_{\mathbf{x}}^{s}}) + \frac{\mathbf{K}_{\mathbf{x},m} \mathbf{K}_{\mathbf{x},m}^{\top}}{\sigma_{g}^{2}}\right)^{-1}\right)$$

$$q(\mathbf{G}_{\mathbf{x}}) = \prod_{i=1}^{N_{\mathbf{x}}} \mathcal{N}\left(\boldsymbol{g}_{\mathbf{x},i}; \Sigma(\boldsymbol{g}_{\mathbf{x},i}) \left(\frac{\widetilde{\mathbf{A}_{\mathbf{x}}^{\top}} \boldsymbol{k}_{\mathbf{x},i}}{\sigma_{g}^{2}} + \widetilde{\mathbf{G}_{\mathbf{z}}} \widetilde{(\boldsymbol{f}^{i})^{\top}}\right), \left(\frac{\mathbf{I}}{\sigma_{g}^{2}} + \widetilde{\mathbf{G}_{\mathbf{z}} \mathbf{G}_{\mathbf{z}}^{\top}}\right)^{-1}\right)$$

$$q(\mathbf{F}) = \prod_{i=1}^{N_{\mathbf{x}}} \prod_{j=1}^{N_{\mathbf{z}}} \mathcal{TN}(f_{j}^{i}; \widetilde{\boldsymbol{g}_{\mathbf{x},i}^{\top}} \widetilde{\boldsymbol{g}_{\mathbf{z},j}}, 1, f_{j}^{i} y_{j}^{i} > \nu).$$

B Kernelized Bayesian Matrix Factorization with Twin Multiple Kernel Learning for Real-Valued Outputs

We modify our proposed model for binary-valued outputs to also handle real-valued outputs. Figure 8 illustrates the graphical model of the modified *kernelized Bayesian matrix factorization with twin multiple kernel learning* (KBMF2MKL) with latent variables and their corresponding priors.

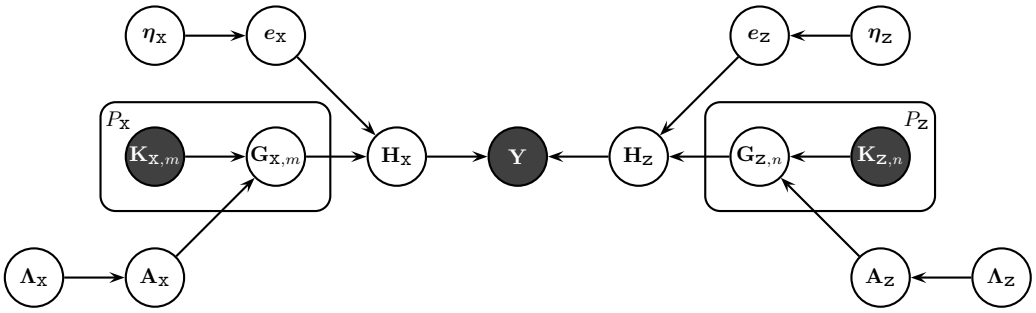

Figure 8: Graphical model of kernelized Bayesian matrix factorization with twin multiple kernel learning for real-valued outputs.

The distributional assumptions of the modified KBMF2MKL model are

$$\begin{aligned}
\lambda_{\mathbf{x},s}^{i} &\sim \mathcal{G}(\lambda_{\mathbf{x},s}^{i}; \alpha_{\lambda}, \beta_{\lambda}) && \forall(i,s)\\
a_{\mathbf{x},s}^{i} | \lambda_{\mathbf{x},s}^{i} &\sim \mathcal{N}(a_{\mathbf{x},s}^{i}; 0, (\lambda_{\mathbf{x},s}^{i})^{-1}) && \forall(i,s)\\
g_{\mathbf{x},m,i}^{s} | \boldsymbol{a}_{\mathbf{x},s}, \boldsymbol{k}_{\mathbf{x},m,i} &\sim \mathcal{N}(g_{\mathbf{x},m,i}^{s}; \boldsymbol{a}_{\mathbf{x},s}^{\top} \boldsymbol{k}_{\mathbf{x},m,i}, \sigma_{g}^{2}) && \forall(m,s,i)\\
\eta_{\mathbf{x},m} &\sim \mathcal{G}(\eta_{\mathbf{x},m}; \alpha_{\eta}, \beta_{\eta}) && \forall m\\
e_{\mathbf{x},m} | \eta_{\mathbf{x},m} &\sim \mathcal{N}(e_{\mathbf{x},m}; 0, \eta_{\mathbf{x},m}^{-1}) && \forall m\\
h_{\mathbf{x},i}^{s} | \{e_{\mathbf{x},m}, g_{\mathbf{x},m,i}^{s}\}_{m=1}^{P_{\mathbf{x}}} &\sim \mathcal{N}\left(h_{\mathbf{x},i}^{s}; \sum_{m=1}^{P_{\mathbf{x}}} e_{\mathbf{x},m} g_{\mathbf{x},m,i}^{s}, \sigma_{h}^{2}\right) && \forall(s,i)\\
y_{j}^{i} | \boldsymbol{h}_{\mathbf{x},i}, \boldsymbol{h}_{\mathbf{z},j} &\sim \mathcal{N}(y_{j}^{i}; \boldsymbol{h}_{\mathbf{x},i}^{\top} \boldsymbol{h}_{\mathbf{z},j}, \sigma_{y}^{2}) && \forall(i,j).
\end{aligned}$$

As short-hand notations, all hyper-parameters in the model are denoted by $\boldsymbol{\zeta} = \{\alpha_{\eta}, \beta_{\eta}, \alpha_{\lambda}, \beta_{\lambda}, \sigma_{g}, \sigma_{h}, \sigma_{y}\}$, all prior variables by $\boldsymbol{\Xi} = \{\boldsymbol{\eta}_{\mathbf{x}}, \boldsymbol{\eta}_{\mathbf{z}}, \boldsymbol{\Lambda}_{\mathbf{x}}, \boldsymbol{\Lambda}_{\mathbf{z}}\}$, and the remaining random variables by $\boldsymbol{\Theta} = \{\mathbf{A}_{\mathbf{x}}, \mathbf{A}_{\mathbf{z}}, \boldsymbol{e}_{\mathbf{x}}, \boldsymbol{e}_{\mathbf{z}}, \{\mathbf{G}_{\mathbf{x},m}\}_{m=1}^{P_{\mathbf{x}}}, \{\mathbf{G}_{\mathbf{z},n}\}_{n=1}^{P_{\mathbf{z}}}, \mathbf{H}_{\mathbf{x}}, \mathbf{H}_{\mathbf{z}}\}$. We again omit the dependence on $\boldsymbol{\zeta}$ for clarity. We can write the factorized variational approximation as

$$p(\boldsymbol{\Theta}, \boldsymbol{\Xi} | \{\mathbf{K}_{\mathtt{x},m}\}_{m=1}^{P_\mathtt{x}}, \{\mathbf{K}_{\mathtt{z},n}\}_{n=1}^{P_\mathtt{z}}, \mathbf{Y}) \approx q(\boldsymbol{\Theta}, \boldsymbol{\Xi}) = q(\boldsymbol{\Lambda}_\mathtt{x}) q(\mathbf{A}_\mathtt{x}) q(\{\mathbf{G}_{\mathtt{x},m}\}_{m=1}^{P_\mathtt{x}}) q(\boldsymbol{\eta}_\mathtt{x}) q(\boldsymbol{e}_\mathtt{x}) q(\mathbf{H}_\mathtt{x}) q(\boldsymbol{\Lambda}_\mathtt{z}) q(\mathbf{A}_\mathtt{z}) q(\{\mathbf{G}_{\mathtt{z},n}\}_{n=1}^{P_\mathtt{z}}) q(\boldsymbol{\eta}_\mathtt{z}) q(\boldsymbol{e}_\mathtt{z}) q(\mathbf{H}_\mathtt{z})$$

and define each factor in the ensemble just like its full conditional:

$$\begin{aligned}
q(\boldsymbol{\Lambda}_\mathtt{x}) &= \prod_{i=1}^{N_\mathtt{x}} \prod_{s=1}^{R} \mathcal{G}(\lambda_{\mathtt{x},s}^i; \alpha(\lambda_{\mathtt{x},s}^i), \beta(\lambda_{\mathtt{x},s}^i)) \\
q(\mathbf{A}_\mathtt{x}) &= \prod_{s=1}^{R} \mathcal{N}(\boldsymbol{a}_{\mathtt{x},s}; \mu(\boldsymbol{a}_{\mathtt{x},s}), \Sigma(\boldsymbol{a}_{\mathtt{x},s})) \\
q(\mathbf{G}_{\mathtt{x},m}) &= \prod_{i=1}^{N_\mathtt{x}} \mathcal{N}(\boldsymbol{g}_{\mathtt{x},m,i}; \mu(\boldsymbol{g}_{\mathtt{x},m,i}), \Sigma(\boldsymbol{g}_{\mathtt{x},m,i})) \qquad \forall m \\
q(\boldsymbol{e}_\mathtt{x}) &= \mathcal{N}(\boldsymbol{e}_\mathtt{x}; \mu(\boldsymbol{e}_\mathtt{x}), \Sigma(\boldsymbol{e}_\mathtt{x})) \\
q(\boldsymbol{\eta}_\mathtt{x}) &= \prod_{m=1}^{P_\mathtt{x}} \mathcal{G}(\eta_{\mathtt{x},m}; \alpha(\eta_{\mathtt{x},m}), \beta(\eta_{\mathtt{x},m})) \\
q(\mathbf{H}_\mathtt{x}) &= \prod_{i=1}^{N_\mathtt{x}} \mathcal{N}(\boldsymbol{h}_{\mathtt{x},i}; \mu(\boldsymbol{h}_{\mathtt{x},i}), \Sigma(\boldsymbol{h}_{\mathtt{x},i})).
\end{aligned}$$

We can bound the marginal likelihood using Jensen's inequality:

$$\log p(\mathbf{Y} | \{\mathbf{K}_{\mathtt{x},m}\}_{m=1}^{P_\mathtt{x}}, \{\mathbf{K}_{\mathtt{z},n}\}_{n=1}^{P_\mathtt{z}}) \geq \mathrm{E}_{q(\boldsymbol{\Theta},\boldsymbol{\Xi})}[\log p(\mathbf{Y}, \boldsymbol{\Theta}, \boldsymbol{\Xi} | \{\mathbf{K}_{\mathtt{x},m}\}_{m=1}^{P_\mathtt{x}}, \{\mathbf{K}_{\mathtt{z},n}\}_{n=1}^{P_\mathtt{z}})] - \mathrm{E}_{q(\boldsymbol{\Theta},\boldsymbol{\Xi})}[\log q(\boldsymbol{\Theta}, \boldsymbol{\Xi})]$$

and optimize this bound by maximizing with respect to each factor separately until convergence. The approximate posterior distribution of a specific factor $\boldsymbol{\tau}$ can be found as

$$q(\boldsymbol{\tau}) \propto \exp(\mathrm{E}_{q(\{\boldsymbol{\Theta},\boldsymbol{\Xi}\} \setminus \boldsymbol{\tau})}[\log p(\mathbf{Y}, \boldsymbol{\Theta}, \boldsymbol{\Xi} | \{\mathbf{K}_{\mathtt{x},m}\}_{m=1}^{P_\mathtt{x}}, \{\mathbf{K}_{\mathtt{z},n}\}_{n=1}^{P_\mathtt{z}})]).$$

The approximate posterior distributions of the ensemble can be found as

$$\begin{aligned}
q(\boldsymbol{\Lambda}_\mathtt{x}) &= \prod_{i=1}^{N_\mathtt{x}} \prod_{s=1}^{R} \mathcal{G}\left(\lambda_{\mathtt{x},s}^i; \alpha_\lambda + \frac{1}{2}, \left(\frac{1}{\beta_\lambda} + \frac{\widetilde{(a_{\mathtt{x},s}^i)^2}}{2}\right)^{-1}\right) \\
q(\mathbf{A}_\mathtt{x}) &= \prod_{s=1}^{R} \mathcal{N}\left(\boldsymbol{a}_{\mathtt{x},s}; \Sigma(\boldsymbol{a}_{\mathtt{x},s}) \sum_{m=1}^{P_\mathtt{x}} \frac{\mathbf{K}_{\mathtt{x},m} \widetilde{(\boldsymbol{g}_{\mathtt{x},m}^s)^\top}}{\sigma_g^2}, \left(\mathrm{diag}(\widetilde{\boldsymbol{\lambda}_\mathtt{x}^s}) + \sum_{m=1}^{P_\mathtt{x}} \frac{\mathbf{K}_{\mathtt{x},m} \mathbf{K}_{\mathtt{x},m}^\top}{\sigma_g^2}\right)^{-1}\right) \\
q(\mathbf{G}_{\mathtt{x},m}) &= \prod_{i=1}^{N_\mathtt{x}} \mathcal{N}\left(\boldsymbol{g}_{\mathtt{x},m,i}; \Sigma(\boldsymbol{g}_{\mathtt{x},m,i}) \left(\frac{\widetilde{\mathbf{A}_\mathtt{x}^\top} \boldsymbol{k}_{\mathtt{x},m,i}}{\sigma_g^2} + \frac{\widetilde{e_{\mathtt{x},m}} \widetilde{\boldsymbol{h}_{\mathtt{x},i}}}{\sigma_h^2} - \sum_{o \neq m} \frac{\widetilde{e_{\mathtt{x},m} e_{\mathtt{x},o}} \widetilde{\boldsymbol{g}_{\mathtt{x},o,i}}}{\sigma_h^2}\right), \left(\frac{\mathbf{I}}{\sigma_g^2} + \frac{\widetilde{e_{\mathtt{x},m}^2} \mathbf{I}}{\sigma_h^2}\right)^{-1}\right) \qquad \forall m \\
q(\boldsymbol{\eta}_\mathtt{x}) &= \prod_{m=1}^{P_\mathtt{x}} \mathcal{G}\left(\eta_{\mathtt{x},m}; \alpha_\eta + \frac{1}{2}, \left(\frac{1}{\beta_\eta} + \frac{\widetilde{e_{\mathtt{x},m}^2}}{2}\right)^{-1}\right) \\
q(\boldsymbol{e}_\mathtt{x}) &= \mathcal{N}\left(\boldsymbol{e}_\mathtt{x}; \Sigma(\boldsymbol{e}_\mathtt{x}) \left[\frac{\widetilde{\mathbf{G}_{\mathtt{x},m}^\top} \widetilde{\mathbf{H}_\mathtt{x}}}{\sigma_h^2}\right]_{m=1}^{P_\mathtt{x}}, \left(\mathrm{diag}(\widetilde{\boldsymbol{\eta}_\mathtt{x}}) + \left[\frac{\widetilde{\mathbf{G}_{\mathtt{x},m}^\top \mathbf{G}_{\mathtt{x},o}}}{\sigma_h^2}\right]_{m=1,o=1}^{P_\mathtt{x},P_\mathtt{x}}\right)^{-1}\right) \\
q(\mathbf{H}_\mathtt{x}) &= \prod_{i=1}^{N_\mathtt{x}} \mathcal{N}\left(\boldsymbol{h}_{\mathtt{x},i}; \Sigma(\boldsymbol{h}_{\mathtt{x},i}) \left(\sum_{m=1}^{P_\mathtt{x}} \frac{\widetilde{e_{\mathtt{x},m}} \widetilde{\boldsymbol{g}_{\mathtt{x},m,i}}}{\sigma_h^2} + \frac{\widetilde{\mathbf{H}_\mathtt{z}} \widetilde{(\boldsymbol{y}^i)^\top}}{\sigma_y^2}\right), \left(\frac{\mathbf{I}}{\sigma_h^2} + \frac{\widetilde{\mathbf{H}_\mathtt{z} \mathbf{H}_\mathtt{z}^\top}}{\sigma_y^2}\right)^{-1}\right).
\end{aligned}$$